Software Modeling for Open Distributed Network Monitoring Systems

Jacob W. Kallman, Pedrum Minaie, Jason Truppi, Sergiu M. Dascalu, Frederick C. Harris, Jr.

Department of Computer Science and Engineering, University of Nevada, Reno
1664 N. Virginia St., Reno, NV, 89557 USA
jkallman@cs.unr.edu, pminaie@ieee.org,
jason@physics.unr.edu, dascalus@cs.unr.edu, fredh@cs.unr.edu

Abstract. As computer networks grow in size, both physically and geographically, more scalable solutions to network administration are becoming necessary. This need is amplified by the spread of faster and more devastating computer viruses. Furthermore, when dealing with partial and intermittent systems, the need for accompanying network mapping and monitoring with efficient mapping visualization becomes even more important. This paper presents the Open Distributed Network Monitoring (ODNM) package, a software tool that proposes a novel solution for dealing with these issues. Emphasizing the value of well defined software requirements, the package addresses the need for scalability and speed by utilizing a distributed scanning capability that divides the network to be scanned into multiple parallel scans. Excerpts from ODNM's software model, including functional and non-functional requirements, use cases, class diagram and prototype screenshots are presented in the paper and the package's goals, progress, and future development are discussed.

1 Introduction

Maintaining a computer network can be a tedious job, especially for large to enterprise-sized networks. Numerous network administration tools exist that help ease the burden on an organization's system administrators' workload. These tools provide network administrators information about a network's security, performance, and overall layout. However, several problems still exist with today's network mapping systems, the most important of them being the following:

- Applications do not accurately map networks. One commercial application simply finds all the devices for a given subnet and it is then the network administrator's job to draw out the physical network map [1];
- Scanning for large to enterprise-sized networks is both cumbersome and inefficient on a single server. Only one out of six commercial and open-source applications researched use a distributed server model to map out the network [1,2,3,4,5,6] (HP's OpenView Extended Network Node Manager supports a distributed architecture [4]);

- Currently available solutions do not give administrators adequate remote access to scanning, nor do they allow real-time automated scanning of networks in a scalable environment. This becomes an extremely important feature when dealing with partial and intermittent systems;
- Commercial applications can be very pricey [4, 5, 6] and there are not many complete open-source solutions that exist for large networks. Of the researched applications, NINO has been found to be the only complete open-source solution [2].

As a proposed solution to these issues, we are currently developing the Open Distributed Network Monitor, or ODNM. This tool is intended to be scalable enough for monitoring large to enterprise-sized networks but it could be used as well for networks in small business or home business environments. It will also be able to work on single LANs to multi-subnet WANs. This is due to its client-server architecture that both allows the client software to be used anywhere in the network and supports a strategical distribution of the server architecture in the network.

By dividing the task of scanning in a modular way and by providing a client interface which can be used across many platforms (including mobile computing solutions) we believe that ODNM can provide a fast scanning utility which can allow remote scan administration and information gathering in close to real-time conditions. The proposed distributed server architecture follows a similar distributed monitoring architecture as discussed by Subramanyan, Miguel-Alonso, and Fortes in [7]. However these authors have proposed using SNMP for their network monitoring and remote node elicitation solution, whereas we are proposing to use ICMP messages, route table information, and other non-SNMP techniques for network monitoring and remote node elicitation.

In order to build ODNM, we have followed a software development process based on a simplified version of the Unified Process (UP) [8] and have employed the Unified Modeling Language (UML) [9, 10] as specification and design notation. In particular, we have relied on the approach and notational guidelines proposed by Arlow and Neustadt in [11]. We have found that by applying a rigorous, systematic, yet efficient software engineering approach many of the tool's requirements as well as its architectural elements have been identified in a timely and precise manner. This, we believe, is particularly useful for tools dealing with partial and intermittent systems, where efficient network mapping and monitoring needs to be accompanied by fast mapping visualization.

The first version of ODNM, currently in its implementation phase, is expected to be ready by the summer of 2004. The inclusion of a number of extensions is planned for this fall and work on the system and its practical application is envisaged to continue until at least the end of 2004.

This paper provides details of the ODNM's software specification, presents preliminary testing results, and outlines directions of future work. In its remaining part, the paper is organized as follows: Section 2 presents the functional requirements of the system, Section 3 shows several of the system's non-functional requirements, Section 4 provides the system's use case diagram and an examples of use case, Section 5 describes ODNM's high level design and includes prototype screenshots and code module explanations, Section 6 reports on preliminary testing and discusses

future development goals, and Section 7 concludes the paper with a summary of the system's most distinguishing characteristics and its potential for future enhancements.

2 Functional Requirements

Before starting the modeling of the ODNM software, we have identified a series of functional and non-functional requirements that need to be satisfied by the application. The present section provides details of the system's functional requirements while Section 3 shows several of the system's more important non-functional requirements.

The style used for presenting these requirements is the practical, efficient one proposed in [11].

In the following, requirements are classified according to three levels of priority: high (level 3), medium (2), and low (1). These levels of priority designate the importance of certain features, both functional and non-functional, that need to be incorporated in ODNM. The highest priority denotes requirements that must be available for a full working version of the application. Medium and low priorities denote requirements that are optional for a full working version of the application but should be considered for more advanced versions of the tool.

2.1 Client Highest Priority (3)

These requirements represent the base requirements that must be met by the client side of the ODNM system:

- The client user shall have the ability to view the network topology either graphically or in a tree-like structure.
- The client interface shall be a simple, yet effective GUI that shall display all important system components designated by the user. The interface shall be tailored to users of all skill levels.
- The client software shall output the completed statistics on a host machine in a simple and relevant format.
- The client user shall have the option to cancel a scan but the GUI shall display devices scanned prior to canceling.
- The node-to-node connection speed shall be displayed between a specified client and the server currently connected to in a client side dialogue.
- The client user shall have the option to select a range of IP's or a subnet given a subnet mask (restricted to Class C subnets).
- The results of a given scan shall be available in two modes: simple and advanced. The simple mode shall provide hostname and IP address and the advanced mode shall provide more detail such as open ports, connection speed, and so forth.
- The client GUI shall be able to manually initiate a server scan or view the results of an automated scan.

- In the graphical structure view, various components shall have a distinctive icon on the map. For example, a printer shall have a printer icon and a computer shall have a computer icon.

2.2 Client Medium Priority (2)

These requirements are optional for the first release, but should be addressed in short-term future work:

- The client software should have a tool to generate reports to HTML or XML.
- The client user should have the ability to export the network map to some sort of image (i.e., JPEG, TIFF, or BMP).
- The client user should be able to view and set the scanning schedule.
- The client user should be able to view an estimated network topology to a certain level of accuracy.
- The user should be able to view the network topology using a physical structure, such as a map of the city of Reno, Nevada, including all transmission lines.

2.3 Server Highest Priority (3)

These requirements represent the base requirements that must be met by the server side of the ODNM system:

- The server shall have the ability to remotely detect the operating system on desired hosts.
- The server shall have the ability to scan all 65,000 TCP/UDP ports on desired hosts.
- The server shall store log files of client-server and server-server communications.
- The server shall be able to run a ping scan on a single host or a range of hosts.
- The server shall relay information back to the client for every new host detected.
- The server shall contain a saved or most recent snapshot of all hosts scanned after every scheduled scan.
- The server shall detect IP addresses (IPV4), open TCP/UDP ports, host operating system, and host name.
- The server shall be able to run scheduled scans and send back reports to client.
- The server shall be configured by reading a configuration file.
- The server shall know of other ODNM servers by the configuration file.

2.4 Server Medium Priority (2)

These requirements are optional for the first release, but should be pursued in short-term future work. An example of such requirements is the following:

- The server should be able to complete baseline and subsequent comparisons of networks to determine any addition or removal of devices in the network (i.e., intrusion detection).

3 Non-Functional Requirements

The most relevant non-functional requirements for DuffNM are listed below.

3.1 Non-Functional Highest Priority (3)

The following represent the base non-functional requirements that must be satisfied by the ODNM system:

- The system shall have the ability to export to HTML and XML documents.
- The system shall have the ability to export network maps in PNG, JPEG, BMP, and TIFF formats.
- The system shall be written in Java and Perl.
- The scan speed shall be reasonably efficient in the distributed environment.
- The system shall have more simplicity then other network scan tools such as HP's OpenView and Ipswitch's WhatsUp Gold software.
- The server shall run in a UNIX environment.
- The client shall run in a Windows or Linux/UNIX environment.

3.2 Non-Functional Medium Priority (2)

The following are examples of non-functional requirements that are optional for the first release, but should be pursued in short-term future work:

- The system's client-server communications should be secure and encrypted.
- The system should have the ability to create reports in Microsoft Word format.

4 Use Cases

Early in the modeling process, the system's functionality has been defined using use cases and scenarios. The entire functionality of the ODNM tool is represented in the use case diagrams shown in Figure 1 (client side) and Figure 2 (server side). A correspondence between the functional requirements listed in Sections 2 and 3 of this paper and the use cases shown in the system's use case diagrams was established for software development purposes.

Due to the tool's specific characteristics, ODNM's use cases have been grouped into two packages, client side and server side. This division of the system's use cases

into two packages corresponds to breaking down the ODNM software into two main executable structures of the program.

The client use cases deal with activities a user can perform on the client side of this software package. The server side includes two actors, a server administrator and time. The server administrator deals with tasks such as configuring the server while in this application time is the impersonal actor that responds to automatically triggered events by the client side, such as initializing a scan or storing logs about the most recent scan.

The use case diagrams shown in Figures 1 and 2 illustrate the basic ways in which outside actors can interact with the system.

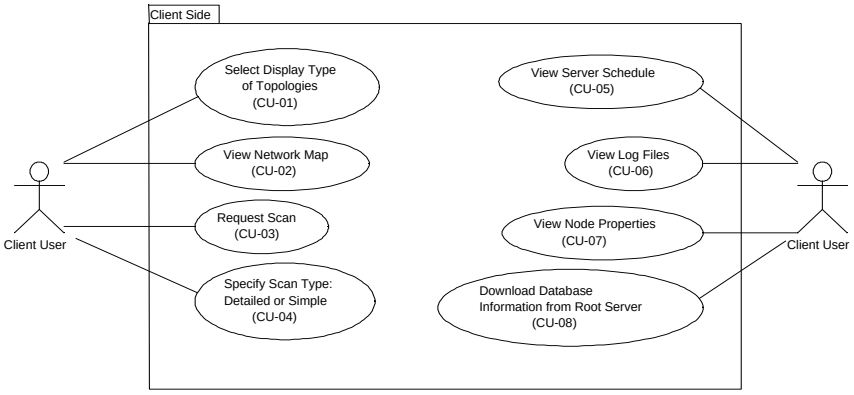

Fig. 1. Client Side Use Case Diagram

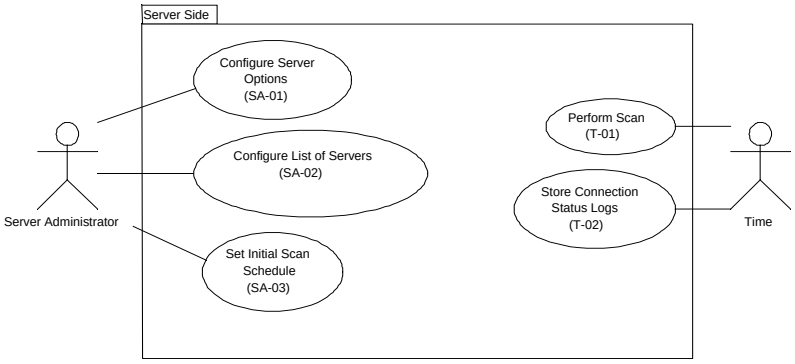

Fig. 2. Server Side Use Case Diagram

Typical to UML-based software specifications, all use cases can be further described using templates such as the one presented in Figure 3 and can be detailed using scenarios. For simplicity, the latter have not be included in this paper but are available in the project's software requirements specification (SRS) document.

Use case: Perform Scan
ID: T-01
Actors: Time
Preconditions: 1 The server is running. 2 There is a valid configuration file
Flow of events: 1 The use case starts when the server either: 1.1 Is run for the first time 1.2 Reaches the time interval at which a scan is to be executed 2 If the database does not exist then the server shall: 2.1 Create a file to store the database in 2.2 Read the configuration file to get the subnets to scan, and the lower level servers 2.3 Scan the subnets listed in the configuration file 2.4 Transfer the information from lower level servers into the database
Postconditions: 1 The database file is updated and closed 2 The time interval counter is reset to zero

Fig. 3. Description of the Perform Scan Use Case

The above excerpts from the tool's software specification have been included in the paper to illustrate the foundation on which the ODNM monitoring software package has been developed. Next, we build upon this foundation by presenting the high level design of the software.

5 High Level Design

The design of ODNM has been intended to be simple and easy to understand by all levels of system administrators and hobbyists. In this section the high level system model is presented (Figure 4), together with an excerpt of the class diagram that has been used to define the structure of ODNM software (Figure 5). The section also presents, in Figures 6 and 7, screenshots of client and server outputs.

5.1 High Level System Context Model

ODNM can be seen as a software layer above the operating system and other essential system functions. In the future, we plan to integrate the network monitor's functionality into the embedded operating systems of switches and routers, but at this point in time it is a standalone application that uses Java and Perl. ODNM also uses Nmap, which is a separate application developed by insecure.org [12], a stopgap that provides support for the actual scanning of the machines. As described in Section 6, this will eventually be replaced by other scanning methods that we plan to develop.

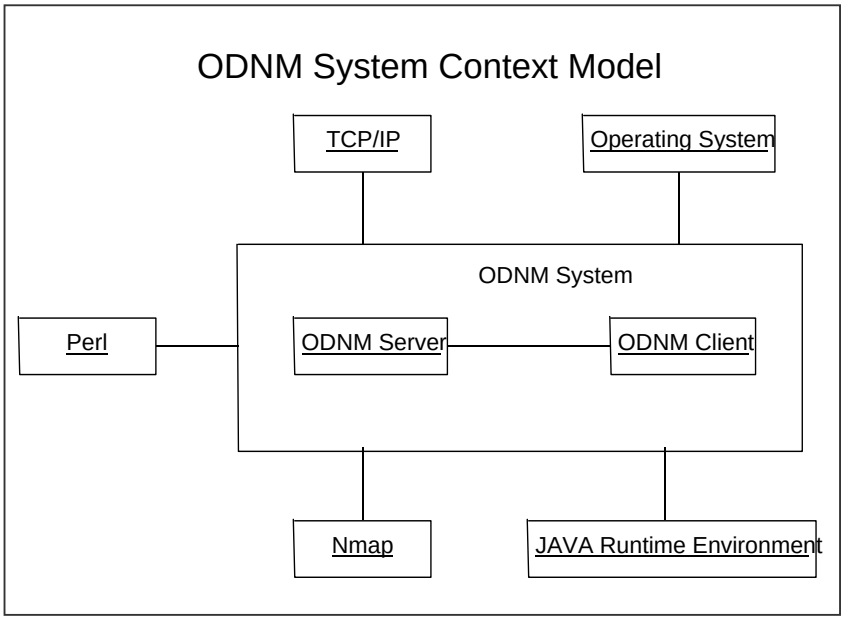

Fig. 4. ODNM High Level System Context Diagram

5.2 Prototype Class Diagram

Due to space limitations, only a part of the class diagram used to design the network monitor's software structure is shown in Figure 5.

5.3 Sample Screenshots

The screenshots shown in Figures 6 and 7 are samples of what ODNM server and client interfaces look like and are intended to give an idea on how the user can interact with them.

The server has been designed to run as a daemon process, without any interaction from the user. It uses a BSD-style configuration file to set server options, and outputs a text file which is human-readable. Other than this, it requires no interaction with the user.

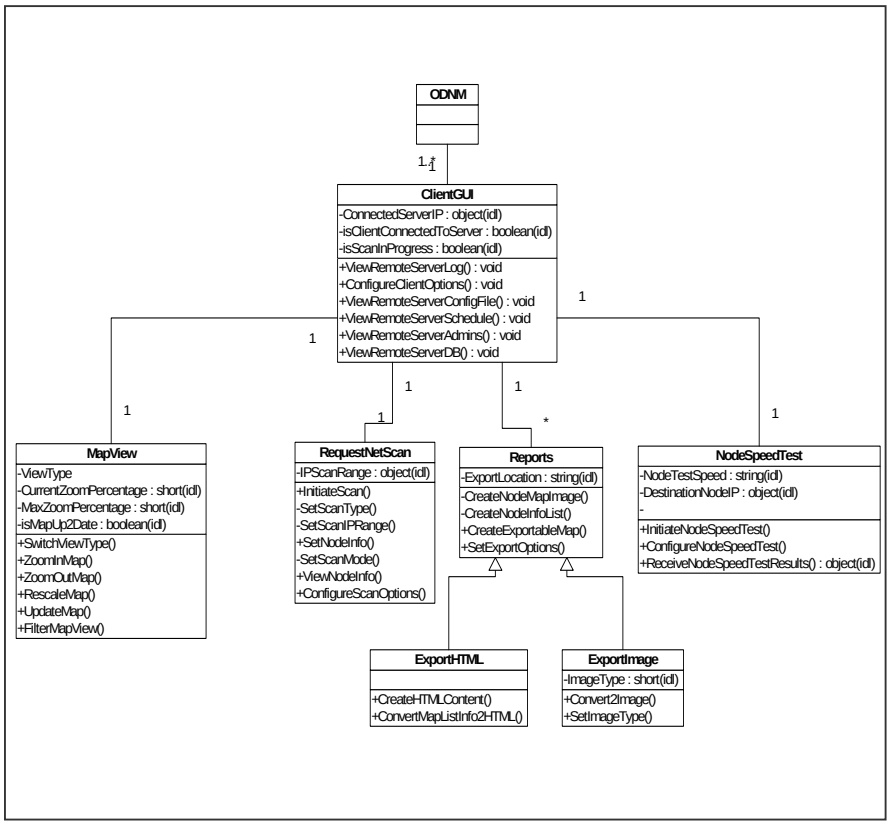

Fig. 5. ODNM Class Diagram (partial)

Figure 6 provides an example of server output. Specifically, it illustrates the data collected by ODNM server – in more detail, it shows the scanned IP address, the speed of the connection (last time/average), and the number of measured hops between the IP and the server scanning, the open ports on the scanned IP, and the detected operating system on the scanned IP address. This is essentially the extent of the server output, but the client has many more user options for the interacting with the system.

IP	Host Name	Hops	Ports	OS
192.168.1.1	router	1	22/tcp/ssh	Linux-2.4.19
192.168.1.2	fs_server	2	NONE	Windows
192.168.1.65	webserver	2	80/tcp/Apache	Linux-2.4.37

Fig. 6. ODNM Server Output Sample

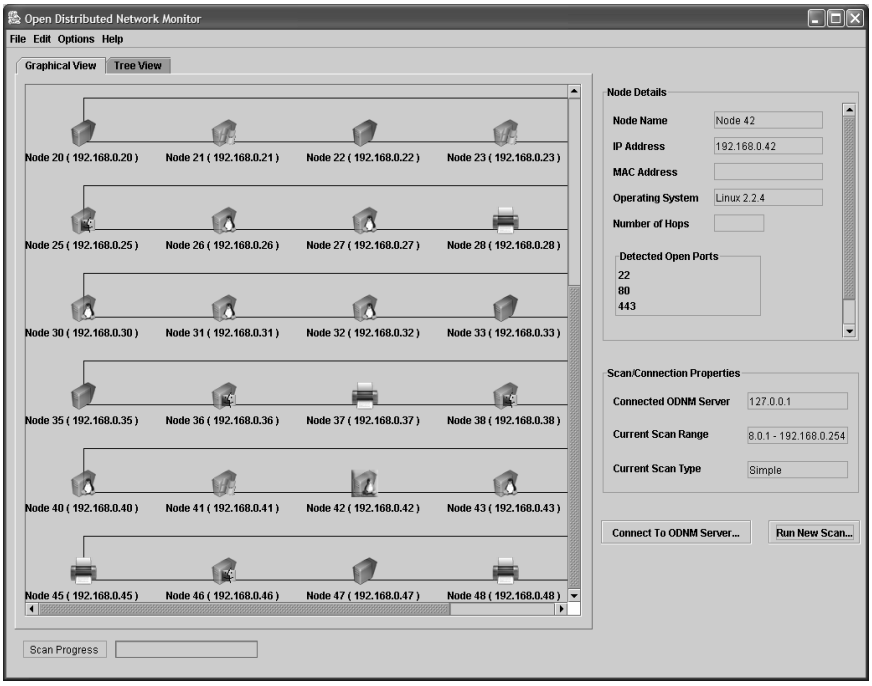

Fig. 7. ODNM Main Client Interface Window

6 Testing and Further Development

An initial, prototype version of ODNM has nearly been completed. The scan module, connection management module, and client interface are all nearing a stage where they are ready to be packaged into an initial release. The main goals that we need to accomplish to reach this stage are as follows:

- Finish refining the client-side user interface and implement topology and physical mapping ability. Currently, a primitive graphical node map has been implemented but as of now it does not compare well with the graphical node representations found in other applications;
- Re-implement the scan module using C/C++ and rebuilt algorithms, instead of simply using Perl and interpreting Nmap output. We hope to use scanning techniques such as remote operating system detection as proposed by insecure.org [13] and topology discovery as proposed by Lowekamp, O'Hallaron, and Gross [14];
- Increase security of the server output by encrypting the information database while on the server and in transit to the client interface;
- Finish development of the database integration module, and make the connection management module more efficient and secure against buffer attacks.

Our preliminary results have been encouraging. In informal tests, it takes about two seconds to scan each machine in a network using the current implementation with Nmap. Theoretically, for a single network monitoring system that uses Nmap, scanning three subnets of two-hundred nodes each may take a minimum of twelve minutes (2 seconds/scan * 3 subnets * 200 devices/subnet = 12 minutes, time for additional overhead and time to access slow subnet connections are not taken into consideration). With the same network configuration, it may take approximately four minutes to scan the network using three ODNM servers placed within each of the three subnets ([2 seconds/scan * 3 subnets * 200 devices/subnet]/3 ODNM servers = 4 minutes, additional overhead not taken into consideration). The time required to scan an entire network using a single server may not seem very relevant, but in an enterprise environment where network monitor servers are monitoring critical devices, including in remote sites, the time to complete scans and notify an administrator of any critical events must be minimized as much as possible to reduce downtime.

We are confident that this initial scan time of approximately two seconds per node will not increase significantly as the project uses Nmap for its preliminary version. Our goal in the near future is to develop our own algorithms, which we hope will run more efficiently and require less time to execute than the current Nmap stopgap and testing code.

7 Conclusion

The ODNM software tool described in this paper can provide a portable, scalable and fast solution to many of today's growing network administration needs. Because it is designed to be distributed and scalable, it can be versatile enough to answer a large variety of networked system design needs. There is a wealth of user options that we plan to integrate into the system, and there is also significant room for developing innovative algorithms and optimization solutions within the general ODNM environment framework.

References

1. WhatsUp Gold (2004). Available as of April 22, 2004 at: http://www.ipswitch.com
2. NINO (2004). Available as of April 22, 2004 at:
3. http://nino.sourceforge.net/
4. PSNMP (2004). Available as of April 22, 2004 at: http://psnmp.sourceforge.net/
5. HP OpenView Network Node Manager 6.4 and Network Node Manager Extended Topology 2.0 (2004). Available as of April 22, 2004 at: http://www.openview.hp.com/
6. NetRadar (2004). Available as of April 22, 2004 at: http://netradar.sourceforge.net/
7. LANSurveyor 8.0 (2004). Available as of April 22, 2004 at: http://www.neon.com
8. Subramanyan, R., Miguel-Alonso, J., Fortes, J.A.: A Scalable SNMP-based Distributed Monitoring System for Heterogeneous Network Computing. Proceedings of the ACM/IEEE Conference on Supercomputing, Dallas, Texas, USA (2000)
9. Jacobson, J, Booch, G., Rumbaugh, J.: The Unified Software Development Process. Addison-Wesley (1999)
10. OMG's UML Resource Page (2004) Available as of April 18, 2004 at: http://www.omg.org/uml
11. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language: User Guide. Addison-Wesley (1998)
12. Arlow, J., Neustadt, I.: UML and the Unified Process: Practical Object-Oriented Analysis and Design. Addison-Wesley (2002)
13. Nmap Security Scanner version 3.50 (2004). Available as of April 15, 2004 at: http://www.insecure.org
14. Nmap (2002): Remote OS Detection via TCP/IP Stack Fingerprinting. Available as of April 21, 2004 at http://www.insecure.org
15. Lowekamp, B., O'Hallaron, D., Gross, T.: Topology Discovery for Large Ethernet Networks. ACM SIGCOMM '01. August 27-31 (2001) Available as of April 15, 2004 at: http://www.acm.org/sigs/sigcomm/sigcomm2001/p19-lowekamp.pdf